Flat Bidirectional Texture Functions

Julien Hadim, Romain Pacanowski, Xavier Granier, Christophe Schlick

▶ **To cite this version:**

Julien Hadim, Romain Pacanowski, Xavier Granier, Christophe Schlick. Flat Bidirectional Texture Functions. [Research Report] RR-7144, INRIA. 2009, pp.24. inria-00440042v2

HAL Id: inria-00440042
https://hal.inria.fr/inria-00440042v2

Submitted on 2 Mar 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Flat Bidirectional Texture Functions

Julien Hadim — Romain Pacanowski — Xavier Granier — Christophe Schlick

N° 7144

December 2009

ISSN 0249-6399 ISRN INRIA/RR--7144--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
BORDEAUX - SUD OUEST

Flat Bidirectional Texture Functions

Julien Hadim [*†] , Romain Pacanowski [*†] , Xavier Granier [*†] ,
Christophe Schlick [*†]

Thème : Perception, cognition, interaction - Interaction et visualisation
Équipes-Projets IPARLA

Rapport de recherche n° 7144 — December 2009 — 21 pages

Abstract:

Highly-realistic materials in computer graphics are computationally and memory demanding. Currently, the most versatile techniques are based on Bidirectional Texture Functions (BTFs), an image-based approximation of appearance. Extremely realistic images may be quickly obtained with BTFs at the price of a huge amount of data. Even though a lot of BTF compression schemes have been introduced during the last years, the main remaining challenge arises from the fact that a BTF embeds many different optical phenomena generated by the underlying meso-geometry (parallax effects, masking, shadow casting, inter-reflections, etc.).

We introduce a new representation for BTFs that isolates parallax effects. On one hand, we built a flattened BTF according to a global spatial parameterization of the underlying meso-geometry. On the other hand, we generate a set of view-dependent indirection maps on this spatial parameterization to encode all the parallax effects. We further analyze this representation on a various set of synthetic BTF data to show its benefits on view-dependent coherency, and to find the best sampling strategy. We also demonstrate that this representation is well suited for hardware acceleration on current GPUs.

Key-words: BTF, Reflectance and Shading Models, Texture Tapping, Meso-structure, Hardware Rendering

* IPARLA Project (INRIA Bordeaux Sud-Ouest - LaBRI)
† {hadim|pacanows|granier|schlick}@labri.fr

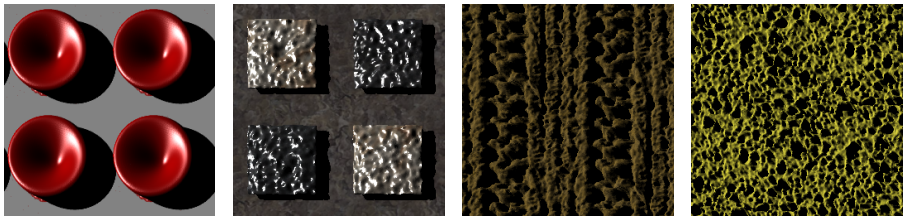
Centre de recherche INRIA Bordeaux – Sud Ouest
Domaine Universitaire - 351, cours de la Libération 33405 Talence Cedex
Téléphone : +33 5 40 00 69 00

Fonctions bidirectionnelles de Texture applaties

Résumé : En Infographie, les matériaux hautement réalistes sont grand consommateurs de puissance de calculs ainsi que de mémoire. A l'heure actuelle, les techniques les plus versatiles reposent sur les fonctions de textures bidirectionnelles (BTFs) représentant une approximation à partir d'images de l'apparence des matériaux. Des images extrêmement réalistes peuvent être obtenues rapidement à l'aide de BTFs au prix d'une énorme quantité de données. Bien que de nombreux schémas de compression de BTFs aient été introduits au cours de ces dernières années, le principal challenge restant provient du fait qu'une BTF mélange différents phénomènes optiques générés par la meso-géométrie sous-jacente (effets de parallaxe ou de masquage, ombres portées, inter-réflexions, etc.), effets qui ne peuvent être que correctement gérés à l'aide d'approches appropriées.

Nous introduisons une nouvelle représentation pour les BTFs qui isole les effets de parallaxes des autres effets. D'une part, nous construisons une BTF aplatie (*flattened*) guidée par une paramétrisation spatiale et globale de la méso-géométrie sous-jacente. D'autre part, nous générons un ensemble de table d'indirections dans cette paramétrisation et pour chaque point de vue, afin d(encoder tous les effets de parallaxe. Nous analysons aussi cette représentation sur un ensemble de BTFs synthétiques afin de montrer l'avantage qu'elle apporte pour la cohérence dépendante du point de vue et pour trouver la meilleure stratégie d'échantillonnage. Nous montrons aussi que cette représentation est particulièrement bien adaptée pour bénéficier de l'accélération matérielle des processeurs graphiques actuels.

Mots-clés : BTF, Réflectance et modèles d'éclairement, méso-structure, rendu sur carte graphique,



Button Isba Pulli Sponge

Figure 1: Overview of the dataset used in our analysis. The Button and the Isba models present some strong features and can not be encoded as height-fields. The Pully and Sponge models are more usual BTFs defined with a height-field meso-geometry. Both of them present close-to-random geometric variations with a strong directional feature for the Pully model. Their meso-geometry are extracted from real BTF [KBD07, MSK07]. The Button and Sponge models present quasi-uniform Phong and diffuse lighting, as the two other models have move spatially variant and random appearance.

1 Introduction

The appearance of a surface is the result of the combined influence of some spatially variant material reflectance and the related meso-geometry of the surface (the meso-geometry encodes all the local geometric deviation from the underlying flat or smooth reference surface). Consequently, a large number of complex optical phenomena may appear according to the lighting and viewing conditions: parallax effects, masking, shadow casting and inter-reflections. All these components could be modeled and rendered specifically, but the resulting computation would be prohibitive, especially with complex reflectance models and highly tessellated meso-geometries. Moreover, acquisition of such elaborated information from real-world materials could be quite difficult. In this context, image-based techniques provide a good and generic trade-off between quality and speed. In recent years, *Bidirectional Texture Function* (BTF) has emerged as a flexible solution for realistic rendering of material with complex appearance.

As opposed to regular 2D textures which usually store appearance parameters that only vary spatially over a surface, BTFs [DvNK99, Dis98] are 6D textures that also store the variation of these appearance parameters when either the viewing or the lighting direction changes. Obviously, this involves a huge amount of data that has to be highly compressed to be manageable. Basically, two main directions for BTF compression have been proposed: either by fitting measured data with standard analytical BRDF models (such as the Lafortune [DLHS01] model, for instance), or by using classical dimension-reduction techniques (such as Principal Component Analysis [KMBK03, LHZ$^+$04], for instance). The results obtained with both approaches are rather disappointing and this lack of efficiency can be mainly explained by the uncorrelated combination of all the complex optical phenomena listed above, which strongly reduces the coherency of the whole data set. For instance, all measured illumination effects in a given area of the underlying meso-geometry do not fall spatially in the same image pixel for each view. To be really efficient, compression techniques have to be aware of these view-dependent variations.

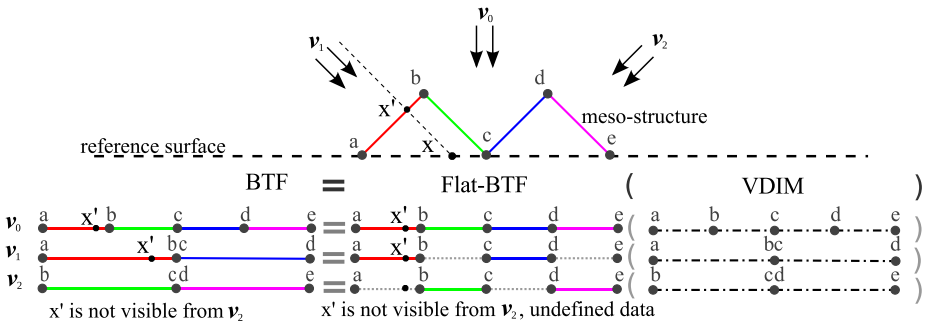


Figure 2: In a standard BTF, the appearance due to the meso-structure is computed or measured independently for a set of view directions $\boldsymbol{v}_0$, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. For each view, the visible illumination is stored but these data are deformed and stretched to fit the corresponding image space (for instance, segment ab is much longer in view $\boldsymbol{v}_1$ than in view $\boldsymbol{v}_0$). On the contrary, in our Flat-BTF model, the samples are stored according to their location on the meso-geometry. Then we use a view dependent indirection map (VDIM) to convert a Flat-BTF into standard BTF representation.

Assuming that a description of the meso-geometry can be provided (like in [KBD07]), the basic idea of our approach is to *flatten* the BTF that is, to *separate the parallax effects from the lighting*. Using a parameterization of the meso-geometry, we encode this parallax effect as view-dependent indirection maps (VDIM) within this parameterization. A nice side effect of this new process is to remove a common limitation of much previous work: *we do not require the meso-geometry to be defined as a height-field* [WWT+03]. The remaining data is then encoded as a parallax-free BTF, that we call *Flat-BTF*, and is much more coherent over the set of viewing directions.

To prove some other advantages of our representation, we perform an analysis on a various set of synthetic BTF data, mainly focusing on the resulting coherency and the reconstruction error. With the coherency analysis, we show that the Flat-BTF representation can be seen as a data preconditioning process that would improve the efficiency of any existing fitting or dimension-reduction technique. With the reconstruction error analysis, we demonstrate that the accuracy of the final rendering can be greatly improved. Finally, we also show that Flat-BTFs are well suited to current graphics hardware, both for the synthesis and the rendering steps.

We introduce our Flat-BTF model as follows. First, we briefly review the large amount of previous work, before describing our representation with its corresponding synthesis and GPU rendering process. Second, we present a comprehensive error study for a various set of characteristic BTF configurations (the dataset is presented in Figure 1). Finally, we discuss some limitations and perspectives for this new representation.

2 Previous Work

The *Bidirectional Texture Function* (BTF), as introduced by Dana *et al.* [DvNK99], is an elegant way to represent materials with complex appearances, by combining many different optical phenomena (parallax, masking, shadow casting, direct shading, inter-reflections, and others) in one single data structure. Generated either by acqui-

sition [MBK05, ND06] or by synthesis [DLHS01], a BTF is basically a sampled 6D function composed of a large set of precomputed images of the material appearance, one for each sample of lighting and viewing directions. The main drawback of the BTF representation is undoubtedly the huge size of the corresponding data structure, which strongly limits its application field.

For a planar local geometry (i.e., when the reference surface is considered as smooth at texture scale and thus does not require an explicit meso-geometry), the appearance can be encoded using a more efficient 6D data structure called *Spatially Variant Bidirectional Distribution Function* (SVBRDF), initially proposed by McAllister *et al.* [MLH02]. As there is no meso-geometry, SVBRDFs embed a much smaller set of visual effects compared to BTFs. Consequently, the corresponding data can be efficiently compressed by using a fitting process based on standard analytical BRDF models [NDM05]. Once fitted, each texel of the resulting texture stores the parameters of the chosen BRDF model. The data compression is thus only performed along the viewing and lighting directions, but no compression is performed spatially. To further compress the dataset, Lensch *et al.* [LKG+03] suggest a hierarchical clustering approach, over the surface. Lawrence *et al.* [LBAD+06] introduce an automatic factorization into a compact shade tree structure.

Similar techniques can be applied for materials with quasi-planar local geometry. However, when the height variation becomes significant, resulting texel values start to include more and more visual effects generated by the meso-geometry: using the terminology proposed in [WHON97], each texel represents an *apparent* BRDF (ABRDF) rather than a true BRDF. Meseth *et al.* [MMK04] and Filip and Haindl [FH04] have shown the limit of traditional BRDF fitting for such complex reflectance fields. They introduce a new representation where a BRDF model is fitted independently for each view to remove the problem of view-dependent coherency. In *Polynomial texture maps*, Malzbender *et al.* [MGW01] show that a simple quadratic form for each view provides convincing results for a large set of materials. But the highest compression rates are obtained using Principal Component Analysis (e.g., [MMK03]).

Ma et al. [MCT+05] propose to separate BRDF and meso-geometry effects by splitting the BTF into a low-pass band (storing an average BRDF of the material) and a highpass band (dealing with meso-geometry effects). The low-pass band is fitted using the standard Phong BRDF, while a PCA is performed on the high-pass band, transformed into a Laplacian pyramid. Vasilescu and Terzopoulos [VT04] introduce a multilinear decomposition for BTF analysis, called Tensor Texture. Haindl and Filip [HF07] reach large compression rates for BTFs by extracting a height-field, a normal map, and by using a statistical representation of the remaining data.

For more accurate rendering, especially at silhouettes, the representation of the surface appearance has to embed some geometric information. Magda and Kriegman [MK06] convert a BTF into a volumetric texture. In the work of Wang *et al.* [WWT+03], the parallax effects are encoded using a view-dependent and curvature-dependent warping texture. However, the technique involves an expensive precomputation step and requires efficient compression algorithms to reduce the size of the resulting data structure. Moreover, this approach is limited to height-field meso-geometries and only accounts for direct lighting.

Relief-mapping is another approach that allows efficient rendering of accurate silhouettes on GPUs. When employing height-field meso-geometries, Badoux and Decoret [BD06] propose to use a binary search combined with precomputed walking steps to find the intersection of the geometry with the viewing direction. Policarpo *et al.* [PO06] extend the relief-mapping technique to more general geometry, using mul-

tiple layers of textures. In their approach, McGuire *et al.* [MW08] introduce a uniform parameterization of the geometry to improve the quality of the relief-mapping texture. All these approaches provide an improved accuracy on silhouettes, but as they involve on-the-fly computation of ray intersection, there is no guaranty on the final rendering frame-rate, unless reducing the computation accuracy. Furthermore, they also require large computation time for the areas that are far away from the silhouettes, where standard BTFs would be sufficient.

To sum up, one can state that all these techniques show that the accuracy and the efficiency strongly rely on the correct approximation of the underlying effects of the meso-geometry. In the solution introduced in this report, we propose to convert a BTF into an alternative representation closer to a SVBRDF, while keeping the parallax effects of the meso-geometry. This can be seen as a *flattening* process which increases the view-dependent coherency in order to improve compression rates of data fitting or dimension reduction techniques.

3 Flat BTFs

3.1 Overview

As said in the introduction, the main advantage of BTFs is to gather many appearance parameters into one global spatially and directionally varying 6D function. Unfortunately, combining many different optical phenomena into one unique value for each position, leads to an uncorrelated dataset that cannot be accurately fit using standard representations. As illustrated in Figure 2, this is mainly due to the fact that some areas of the meso-structure are hidden for some directions, which introduces large gaps for neighboring positions and/or directions. Even worse, a given pixel does not correspond to the same position on the meso-geometry for each view, which strongly reduces the data coherency when changing the viewing direction.

Flat-BTF: Our core idea is to encode the BTF not on the reference surface, but directly on the meso-geometry: this guarantees that every pixel on the new representation corresponds to one unique 3D position. This also totally removes all parallax effects and leads to an improved coherency over the set of viewing directions. We call this parallax-free BTF encoded on the meso-geometry a *Flat-BTF*. The construction of this Flat-BTF is based on the assumption that some kind of representation of the meso-geometry is available: it naturally exists for synthetic BTFs and could be extracted for acquired ones (like in [KBD07]). A Flat-BTF can thus be considered as the lighting equivalent of a geometry image [GGH02] for the underlying geometry.

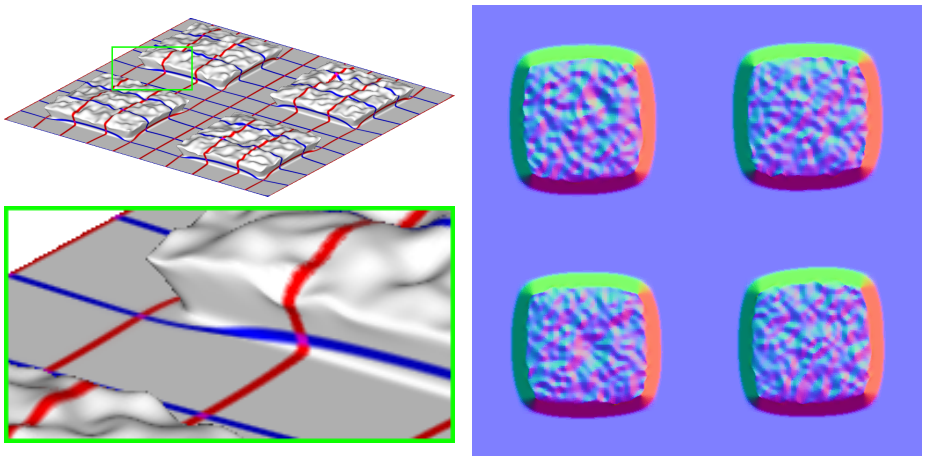
View-Dependent Indirection Map (VDIM): Since the Flat-BTF is linked to the meso-geometry and not to the reference surface that defines the 3D object, it can only be accessed via an indirection map. This indirection is view-dependent and embeds all parallax effects. We call it *View-Dependent Indirection Map* (VDIM). The combination of the Flat-BTF and the VDIM leads to the full reconstruction of a standard BTF (see Figure 2):

$$BTF_{\boldsymbol{v},\boldsymbol{l}}(i,j) = FBTF_{\boldsymbol{v},\boldsymbol{l}}(VDIM_{\boldsymbol{v}}(i,j)), \tag{1}$$

where $\boldsymbol{v}$ is the view direction, $\boldsymbol{l}$ the light direction and (i, j) a pixel position. Note that, even though the Flat-BTF is parallax free, it is still view-dependent since it represents a lighting function.

This map is somehow similar to the view-dependent displacement mapping (VDM) technique presented in [WWT+03]. The main difference is that our VDIM is based on a parameterization of the meso-geometry (see Section 3.2) which solves two limitations of the VDM technique: first, the meso-geometry is not required to be a height-field (see Figure 3), and second, the curvature of the reference surface can be handled more accurately by using this parameterization to remap the meso-structure on a geometric proxy with equivalent curvature.

3.2 Parameterization



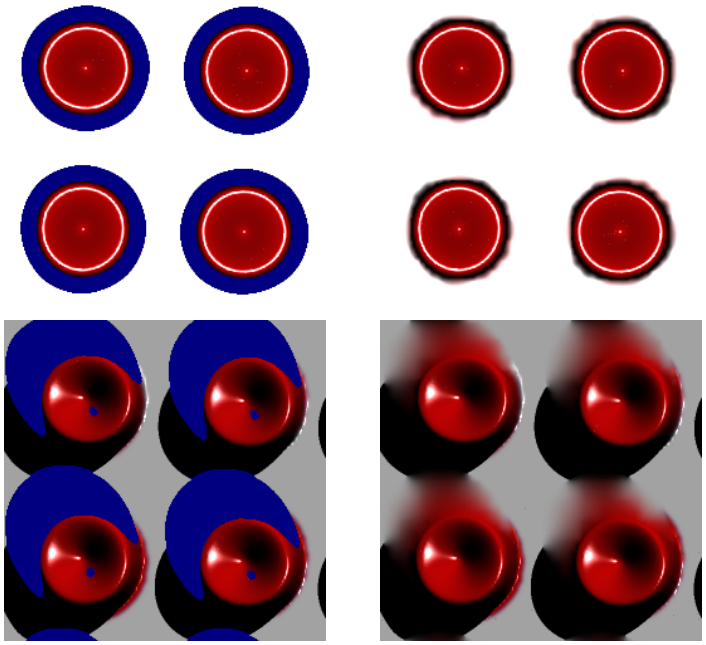
(a) (b)

Figure 3: (a) Parameterization of the meso-geometry for the "Isba" model and, below, a close-up view on a fold (note that the meso-geometry does not correspond to a height-field). (b) The corresponding normal map in the parameterized space shows that all the geometry is correctly represented.

To be able to encode the Flat-BTF, we need to compute a parameterization of the meso-geometry that will provide us with the required indirection between the reference surface and the meso-geometry. As a side effect, this parameterization would also remove the height-field limitation for the shape of the meso-geometry. To accurately preserve all the details, we require the parameterization to be as uniform as possible, in order to preserve the areas of the underlying geometry. In our implementation, we use the *Mean Value Coordinate* method [Flo03] implemented in the Graphite software [Gra03], which is guaranteed to work nicely even for non-manifold surfaces as it is the case here. The result of this parameterization algorithm on the "Isba" model can be seen on Figure 3(a) where blue (resp. red) lines represent the u (resp. v) iso-values. This algorithm has shown to be robust for all tested meso-geometries.

3.3 Flat-BTF Construction

Thanks to the previous parameterization, we can use a three-step approach to build our Flat-BTF. During the first step, we use the GPU to create a set of maps that define the position and the normal on the meso-geometry for each texel of the Flat-BTF. This is trivially done by rendering the meso-geometry, using the parameterization as vertex



(a) (b)

Figure 4: (a) Initial Flat-BTF including holes (in blue) for undefined texel values (b) Final Flat-BTF where holes have been filled with the Push-Pull algorithm.

coordinates and their position and normal as other attributes. Figure 3(b) shows the resulting normal map for the "Isba" model.

Once computed, we use these maps as input to compute the corresponding Flat-BTF on the CPU-side. For a given texel, the 3D position on the map helps in determining the visibility for a given set of viewing and lighting directions. If a texel is not visible from a given viewing direction, it is declared as undefined for that direction (see Figure 2 and blue regions in Figure 4). Similarly, the visibility from the lighting direction determines the direct shadows. For texels which are both visible from the viewing and the lighting directions, the resulting direct illumination is computed according to the corresponding normal and the local BRDF model. Indirect illumination may also be evaluated for this 3D position by adding a simple Monte-Carlo simulation [DBB06]. Finally, note that to obtain seamless patterns, all computations are performed on a periodic repetition of the meso-geometry, similarly to [DLHS01].

The resulting Flat-BTF contains some holes for the undefined illumination values. If a fitting process is later used to reduce the data size, we can simply ignore these holes, as they will not be used by the fitting process. However, for a direct use of the Flat-BTF, we need to fill them. In our implementation, we have used a simple push-pull algorithm to quickly extrapolate some values from the neighbors (see Figure 4). Note that more accurate — but more expensive — solutions like Poisson inpainting [PGB03] are likely to provide better results.

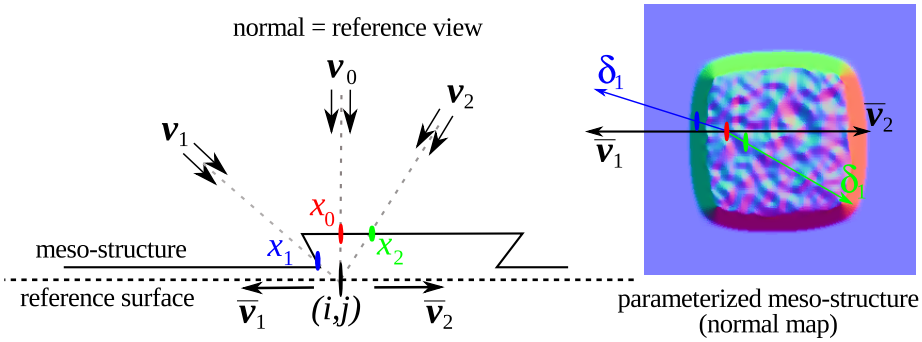


Figure 5: Correlation between the view direction and the displacement of the view-dependent indirection. For each texel (i, j) on the reference plane, and each view $\boldsymbol{v}_k$, we compute the intersection x_k with the meso-geometry. The displacements $\boldsymbol{\delta}_1 = x_1 - x_0$ and $\boldsymbol{\delta}_2 = x_2 - x_0$ on the parameterized space is more or less aligned with the projected view-directions $\bar{\boldsymbol{v}}_1$ and $\bar{\boldsymbol{v}}_2$.

3.4 Construction of View-Dependent Indirection Maps

The computation of the VDIM is also done on the CPU-side. For each texel (i, j) on the reference surface, the intersection with the meso-geometry is computed for each view-direction $\boldsymbol{v}$. The coordinates of that intersection point in the parameter space are then stored at the corresponding map value $VDIM_{\boldsymbol{v}}(i, j)$. As shown in Figure 5, the displacement in the parameter space is more or less aligned with the projection of the view-direction on the reference plane.

For an accurate interpolation between reference positions, we encode our VDIM as a reference indirection $R(i, j)$ and a view-dependent displacement $\Delta_{\boldsymbol{v}}(i, j)$ on the parameterization (as illustrated in Figure 6):

$$VDIM_{\boldsymbol{v}}(i, j) = R(i, j) + \Delta_{\boldsymbol{v}}(i, j)\,. \tag{2}$$

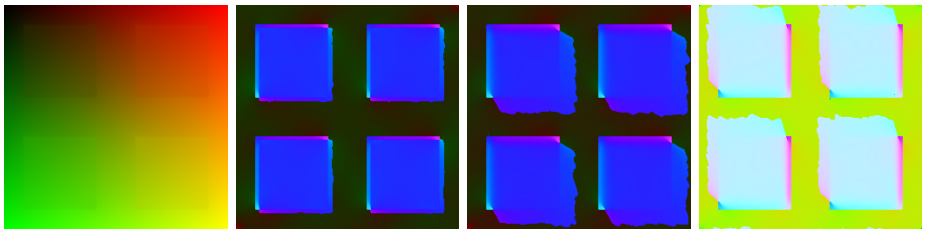


Figure 6: Reference indirection for the normal view for the Isba model and different view-dependent displacement Δ for different view-directions.

Moreover, we encode the displacement Δ as the product of a direction $\boldsymbol{\delta}$ and a distance. Thus, a simple linear interpolation is sufficient for the distance, and a specific interpolation is used for the direction. Since the displacement is roughly aligned with the projection of the view-direction, we choose as reference the indirection for a view direction oriented along the normal of the reference surface $\boldsymbol{n}$:

$$VDIM_{\boldsymbol{v}}(i, j) = VDIM_{\boldsymbol{n}}(i, j) + d_{\boldsymbol{v}}(i, j)\,\boldsymbol{\delta}_{\boldsymbol{v}}(i, j)\,. \tag{3}$$

A nice consequence of that choice is that $d_v = 0$ when $\boldsymbol{v} = \boldsymbol{n}$. This means that $\boldsymbol{\delta_v}$ is quite similar to the projection of $\boldsymbol{v}$ on the reference surface, and that d_v is always increasing. These properties allow a good quality for the reconstruction, as it reduces oscillations when moving around a given position.

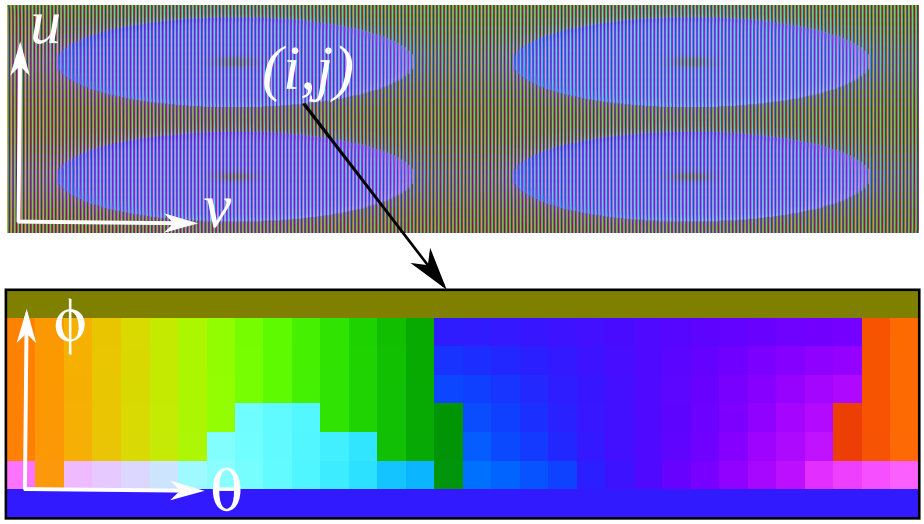


Figure 7: Pixel organization of VDIM used for GPU rendering: for each pixel, we pack the values of $\Delta_v(i,j)$ for the 8×32 directions. Upper image: the whole 2D texture. Lower image: the different view-dependent values for a given pixel. Note: the pole is repeated for the first line.

3.5 Rendering

For interactive rendering, we first need to compute the indirection from the VDIM in order to access to the Flat-BTF. As we demonstrate in Section 4.2, a 16×16 sampling strategy (16 for the zenithal angle ϕ, 16 for the azimuth angle θ) offers an accurate reconstruction of the indirection. Once the indirection maps for each view have been computed, we pack them in one unique 2D texture, as shown in Figure 7. With this packing, we can take advantage of the hardware linear blending between neighboring views.

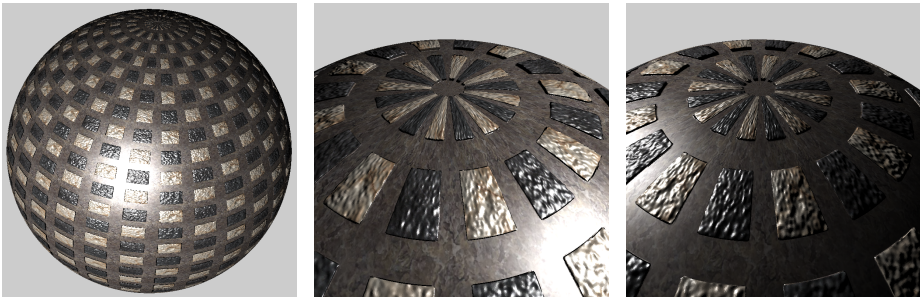
To compute the coordinates (u,v) in the packed texture, we first have to estimate the angles (ϕ,θ) from the view direction $\boldsymbol{v}$. This is done by projecting $\boldsymbol{v}$ onto a local tangent frame. Then the new coordinates (u,v) are computed from the original texture coordinates (i,j) as follows:

$$u = \frac{1}{N}\left(\lfloor i * N \rfloor + \alpha\frac{2\phi}{\pi} + \beta\right)$$
$$v = \frac{1}{M}\left(\lfloor j * M \rfloor + \alpha\frac{\theta}{2\pi} + \beta\right),$$

where $N \times M$ is the unpacked texture resolution, and (α,β) are set up to guarantee that no interpolation can occur in-between two neighboring pixels. Thanks to this approach, the overhead of the VDIM is reduced.

The view-independent reference map is accessed using the original texture coordinates (i, j), without any filtering. Enabling spatial filtering here would be incoherent with the resulting spatial filtering of the packed texture.

Finally, for interactive rendering on GPU, the size of the Flat-BTF has to be reduced. Several techniques may be employed for that purpose. In our implementation, we use a simple representation combining a normal map and a diffuse color texture.



(a) (b) (c)

Figure 8: 3D rendering using our VDIM implementation on GPU (NVIDIA GeForce 8800 GTX) where the Flat-BTF is simply defined by a normal map and a diffuse color texture. For a 800×800 resolution, the frame-rate is around 150 fps. (a) Global view: the parallax effects create an overall feeling of relief. (b) and (c) A closer look for two other lighting directions.

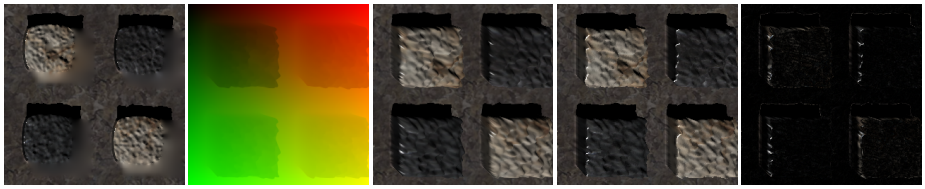
Since we make highly use of the fragment shader, the resulting frame-rate depends on the image resolution. For the close view of Figure 8, we are able to render the 3D object at about 150 frames per second for a 800×800 resolution. In average for such a resolution, the frame-rate is above 200 (cf. Figure 12).

4 Error Study

For our experimentation, we have generated a set of BTFs and their corresponding Flat-BTFs with various shapes of meso-geometries and various material properties (see Figure 1). Each model has been computed 81×81 viewing and lighting directions, with a spatial resolution on 256×256 texels. This is consistent with the sampling strategy used in the BTF database available from the Bonn University (*http: //btf.cs.uni-bonn.de*). All the generated BTFs and Flat-BTFs include direct illumination as well as casted shadows, but no indirect illumination as the corresponding simulation process is quite time consuming, although not introducing noticeable differences in our error study.

Name	Sponge	Button	Isba	Pully
Average	10.23	3	2.91	5.46
Max	15.10	8.17	9.56	9.12
Variance	0.63	0.46	2.41	0.30

Table 1: Average, maximum and variance of texel difference (in Lab distance) between reference BTF and reconstructed one from the equivalent Flat-BTF representation.



(a) (b) (c) (d) (d)

Figure 9: (a) Flat-BTF (b) VDIM (c) Reconstructed BTF slice (d) Reference BTF slice (c) Absolute RGB difference.

4.1 Reconstruction error

Standard BTF textures can be obtained by combining Flat-BTF and VDIM for all sampled views. However, the sampling of a Flat-BTF is uniform on the parameterized meso-geometry, whereas it is uniform on the reference surface for a standard BTF. This different sampling strategies may lead to sampling error and aliasing during the reconstruction. Before any further investigation, we first check whether the difference between the synthesized BTF and the one reconstructed from the Flat-BTF is visually noticeable. To reduce the potential aliasing effects, we generate all the VDIMs by oversampling and use linear filtering of neighboring Flat-BTF texels during the reconstruction.

Figure 9 shows this reconstruction error and Table 1 presents the resulting average difference in the Lab space. As shown in Table 1, the visual error generated over the view-directions is not noticeable. The maximal difference is obtained for the " Sponge" model, mainly because of its highly varying meso-geometry. Moreover, as illustrated in Figure 9, the spatial error only introduces a displacement of roughly 1 texel.

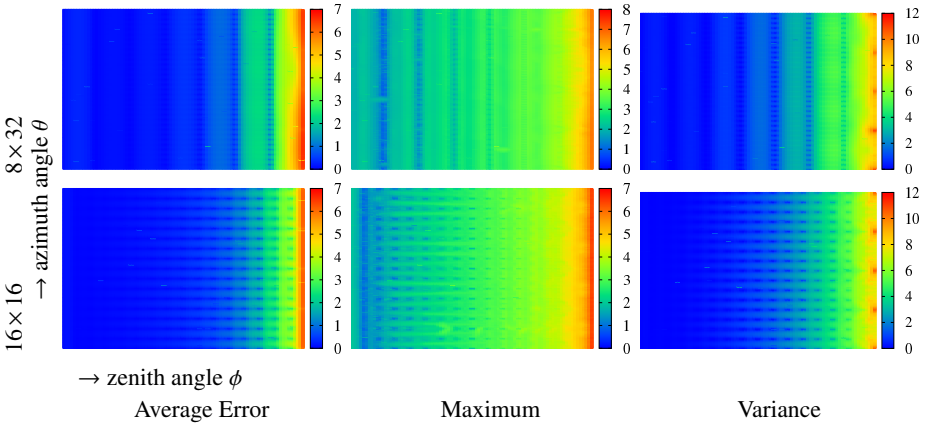


Figure 10: Indirection error on the "Button model" for two sampling strategies: 8×32 and 16×16. The error is computed for a given viewing direction and averaged over all the texels and lighting directions.

4.2 Interpolation error for the indirection

Since the indirection stored in the VDIMs is interpolated between neighboring sampled views, we have to guarantee the accuracy of this process. For this error analysis, once again we generate the VDIMs by oversampling and compare the resulting interpolated indirection using either a 8×32 or 16×16 sampling strategy. As shown in Figure 10, more samples are required for the zenith angle ϕ than for the azimuth angle θ. This is mainly due to the fact that we use directional interpolation over θ (interpolation on the direction of displacement) and a simple linear interpolation over ϕ (interpolation on the distance). Moreover, as observed with any BTF model, the error becomes significantly larger for near-tangent direction where the BTF assumption fails. This can be reduced using more samples for θ, but never removed.

Name	Sponge	Button	Isba	Pully
Average Variance				
BTF	39.82	35.38	14.29	23.20
FBTF (with filling)	18.95	16.87	8.75	7.83
FBTF (only visible)	5.19	7.14	5.12	1.38
Maximal Variance				
BTF	41.57	41.93	33.14	26.85
FBTF (with filling)	30.32	32.92	29.65	19.46
FBTF (only visible)	24.8	18.89	9.96	10.01

Table 2: Average and maximal variance in Lab color space (81 viewing and lighting directions, 256×256 spatial resolution). We compare original BTF with Flat-BTF either by using our push-pull filling algorithm (see Section 3.3) or by simply removing the undefined texels for the variance estimation.

4.3 View-dependent Coherency

As recalled in Section 2, most of the problems encountered when fitting or compressing BTFs arise from the fact that parallax effects are combined with illumination ones, which often requires to compress the dataset view-by-view [WHON97]. As we want to demonstrate that such approach is not necessary with our Flat-BTF representation, we analyze the coherency of the appearance over the set of viewing directions.

For a given view $\boldsymbol{v}_k$, and a given texel (i, j), we compute a vector $\boldsymbol{L}_k(i, j)$ containing all the illumination values for each lighting direction $\boldsymbol{l}_m$ that is, for BTFs

$$\boldsymbol{L}_k(i, j) = \left\{ BTF_{\boldsymbol{v}_k, \boldsymbol{l}_m}(i, j), \forall m \right\},$$

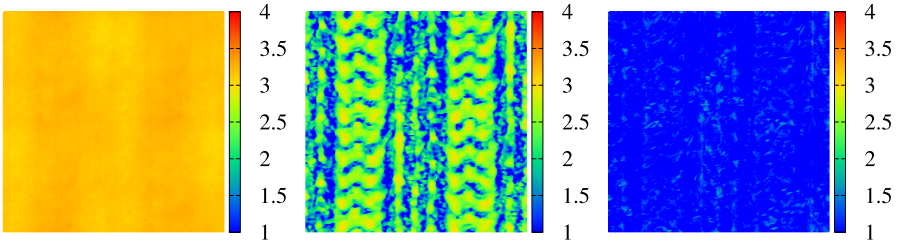
or for Flat-BTFs

$$\boldsymbol{L}_k(i, j) = \left\{ FBTF_{\boldsymbol{v}_k, \boldsymbol{l}_m}(i, j), \forall m \right\}.$$

For each texel, we compute the variance over the set of viewing directions of this vector which provides a measure of the coherency of the appearance. The average variances over the texels are summarized in the Table 2.

As expected, our Flat-BTF presents a much better view-dependent coherency. For all tested models, the average and maximum variance are reduced by at least a factor 2. When using a fitting process, only the values that are actually visible from a given view direction are pertinent (i.e., the blue areas in Figure 4 can be ignored). In this context,

the variance is even further reduced, up to a factor 3, which is likely to greatly improve the quality of such a fitting process. By using better extrapolation techniques than the push-pull algorithm (e.g, Poisson inpainting), the variance of the filled Flat-BTF is also likely to be reduced. We are currently experimenting such an improvement. As a proof of the benefits of our representation, we compress ABRDFs images with standard image file formats *PNG* and *JPEG*. The Table 3 shows that with our representation, ABRDFs can be reduced up to a factor 5.



(a) (b) (c)

Figure 11: Variance of lighting over the viewing directions (Lab distance in logarithmic space) for the "Pulli" model. (a) Variance for a classic BTF. (b) Variance for the corresponding Flat-BTF with undefined texels filled-up. (c) Variance for the Flat-BTF but without taking into account undefined texels.

5 Limitations and perspectives

The main limitation of the approach introduced here is the assumption that the description of the meso-geometry can be provided. Even if an a-priori knowledge on the meso-structure is often used in BTF editing [KBD07, MSK07] or synthesis [TZL+02], its acquisition is not a trivial task for real-world data. Existing solutions mostly rely on shape-from-shading or shadow analysis techniques [DD98, FCM+08] but they only offer a limited precision. In our case, the accuracy of the reconstructed geometry will have an influence on the construction of the VDIM and thus on the related Flat-BTF. This process has to be further investigated. One solution would be to combine the BTF acquisition with a geometric acquisition of the meso-structure, such a combined acquisition would be quite affordable with the recent generation of laser range scanners. Another solution would be to construct the VDIM using similarities in lighting, built on the work presented in [MSK06, ND06].

Another limitation of the approach is the fact that the shadows (i.e., visibility from light directions) are directly integrated into the representation of the Flat-BTF. While the resulting visibility discontinuities are less harmful than the parallax effects which are addressed by our representation, it would nevertheless by interesting to remove them to further improve the global coherency. The solution proposed in the VDM technique [WWT+03] can not be directly applied and needs to be adapted to take into account the deformation due to the parameterization of the meso-geometry.

Our main goal in this report was to prove that our Flat-BTF representation efficiently improves the coherency of the initial BTF dataset. The conversion to a Flat-BTF representation can thus be seen as a preprocess acting as a data preconditioning step. Obviously, the next step is to feed this improved dataset to some standard BTF compression techniques, either based on analytical fitting or on dimension reduction

	PNG Size in Kb		JPEG Size in Kb	
ABRDF	Texel 1	Texel 2	Texel 1	Texel 2
Button	3.3 < 3.4	3.2 < 5.2	10.6 < 12.6	10.6 < 14.1
Isba	2.1 < 4.8	2.5 < 5.7	5.3 < 6.5	7.9 < 11.2
Pully	3.8 < 5.6	1.4 < 5.8	8.9 < 12.9	3.7 < 12.8
Sponge	3.5 < 5.6	1.1 < 5.9	9.2 < 14.7	3.4 < 14.8

Table 3: This table shows the sizes of ABRDF images in kilo-bytes for ABRDFs of Figures 13 and 14 (pages 17 and 18) with two file formats *PNG* and *JPEG* at maximum quality. The number of the concerned ABRDF's texel in the above figures is referenced by the row named "ABRDF". In the inequalities, the left member is the size for the *FBTF* whereas the right member is the size for the classic representation. For some ABRDFs of the models *Pulli* or *Sponge*, the size for the *FBTF* representation is up to 4 or 5 times reduced.

techniques, as recalled in Section 2. However, we have not directly experimented existing BTF compression methods, because we are not convinced that they represent the best solution to compress Flat-BTFs. Indeed, all existing BTF compression techniques are based on the statement that the view-dependent coherency in the dataset is very low. For that reason, most techniques either only try to compress the data view by view, or give a relatively low weighting when fitting data variation for neighboring views. Thanks to our approach, parallax effects are almost totally canceled, so the view coherency is now almost as high as light coherency. As a consequence, there is clearly some room to develop more powerful compression techniques that take this improved coherency into account. We are currently working on that specific point.

6 Conclusion

In this report, we have introduced a new BTF representation, called Flat-BTF, which is intended to isolate parallax effects from illumination effects. The core idea is to use a global spatial parameterization of the meso-geometry, to encode the Flat-BTF on that meso-geometry rather than on the underlying reference surface as done in standard BTF representation. The link between the reference surface and the meso-geometry is stored in a set of View-Dependent Indirection Maps that efficiently encode all parallax effects. As the whole representation can be efficiently stored in a small set of 2D textures, it is particularly well-suited for realtime GPU-based rendering.

We have provided an error analysis on a set of synthetic BTFs showing various shapes of meso-geometries and various material appearances. Compared to standard BTFs, the new representation does not introduce significant visual differences while being much more compact. Furthermore, the most interesting feature is that Flat-BTFs greatly improve the view-dependent coherency and thus would be much better suited for higher compression schemes. We are currently developing such a compression scheme, specifically adapted to this new representation.

Figure 12: Visualization of our Flat-BTF dataset on a collection of 3D objects. The rendering resolution is 800×800 and the average frame rate above 150 fps. First row: the four Flat-BTFs on a bottle. Second row: "Isba" and "Sponge" models on the teapot, for two different lighting directions. Third row: "Button" and "Pully" models on a Vase, for two different viewing directions.

References

[BD06] Lionel Baboud and Xavier Décoret. Rendering geometry with relief textures. In *Proc. Graphics Interface*, pages 195–201. Canadian Information Processing Society, 2006.

[DBB06] Philip Dutré, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., 2006.

[DD98] M. Daum and G. Dudek. On 3-d surface reconstruction using shape from shadows. In *CVPR '98: Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 461, 1998.

[Dis98] Jean-Michel Dischler. Efficiently rendering macro geometric surface structures with bi-directional texture functions. In George Drettakis and Nelson L. Max, editors, *Rendering Techniques*, pages 169–180. Springer, 1998.

[DLHS01] Katja Daubert, Hendrik P. A. Lensch, Wolfgang Heidrich, and Hans-Peter Seidel. Efficient Cloth Modeling and Rendering. In *Proc. EUROGRAPHICS Workshop on Rendering Techniques*, pages 63–70. Springer-Verlag, 2001.

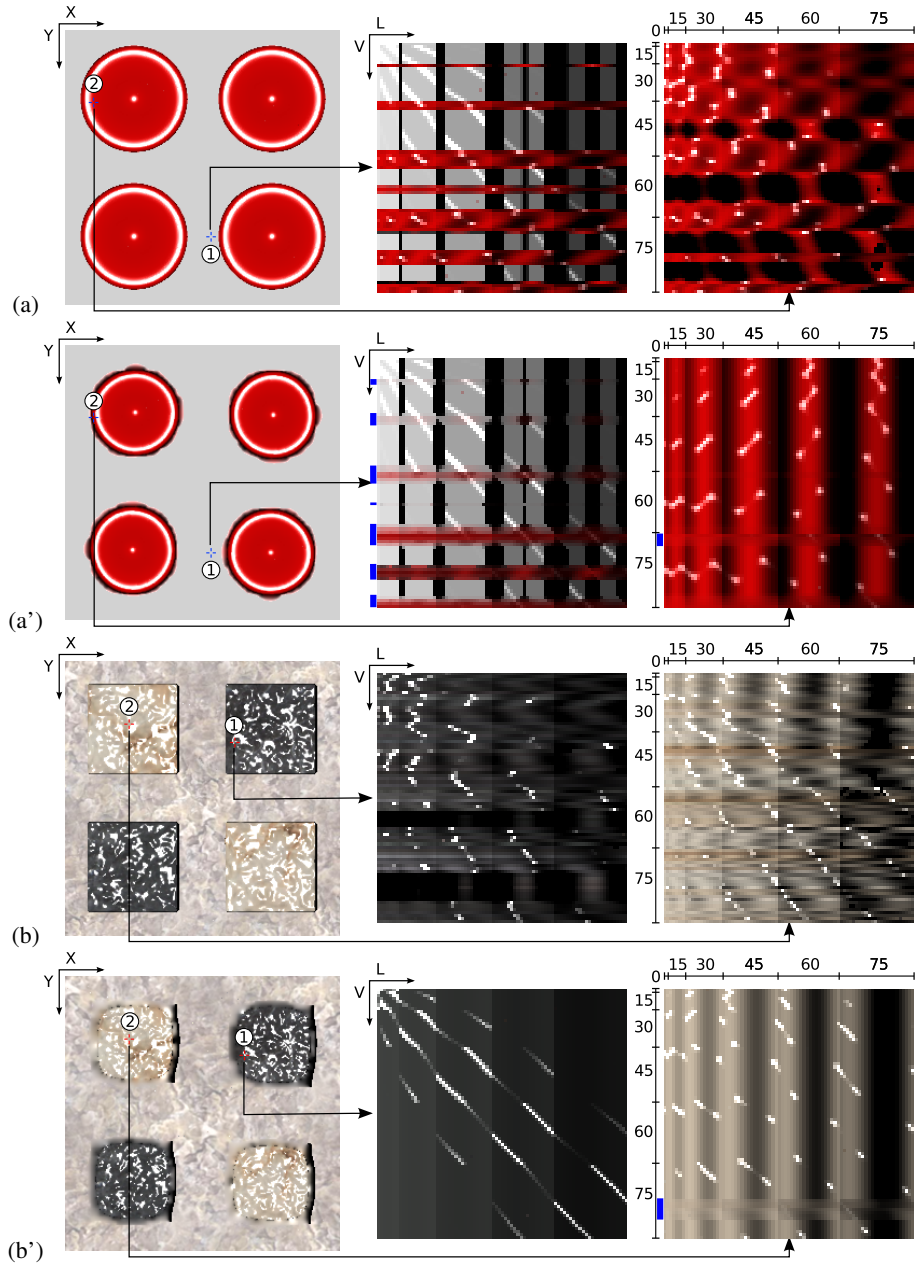


Figure 13: A visual comparison of tipical ABRDFs between a *BTF* and a *FBTF* for the *Bouton* (upper) and *Isba* (lower) models. In (a) and (b), two ABRDFs of the normal *BTF* representation with their localisation in the spatial domain. In (a') and (b'), the corresponding ABRDFs for the *FBTF* function. For each ABRDF pair on a row, the ABRDF (1) has been picked to correspond to the same position on tne underlying meso-structure, while the ABRDF (2) corresponds exactly to the same texel in the image. The blue marks point undefined texels filled by the push-pull algorithm.

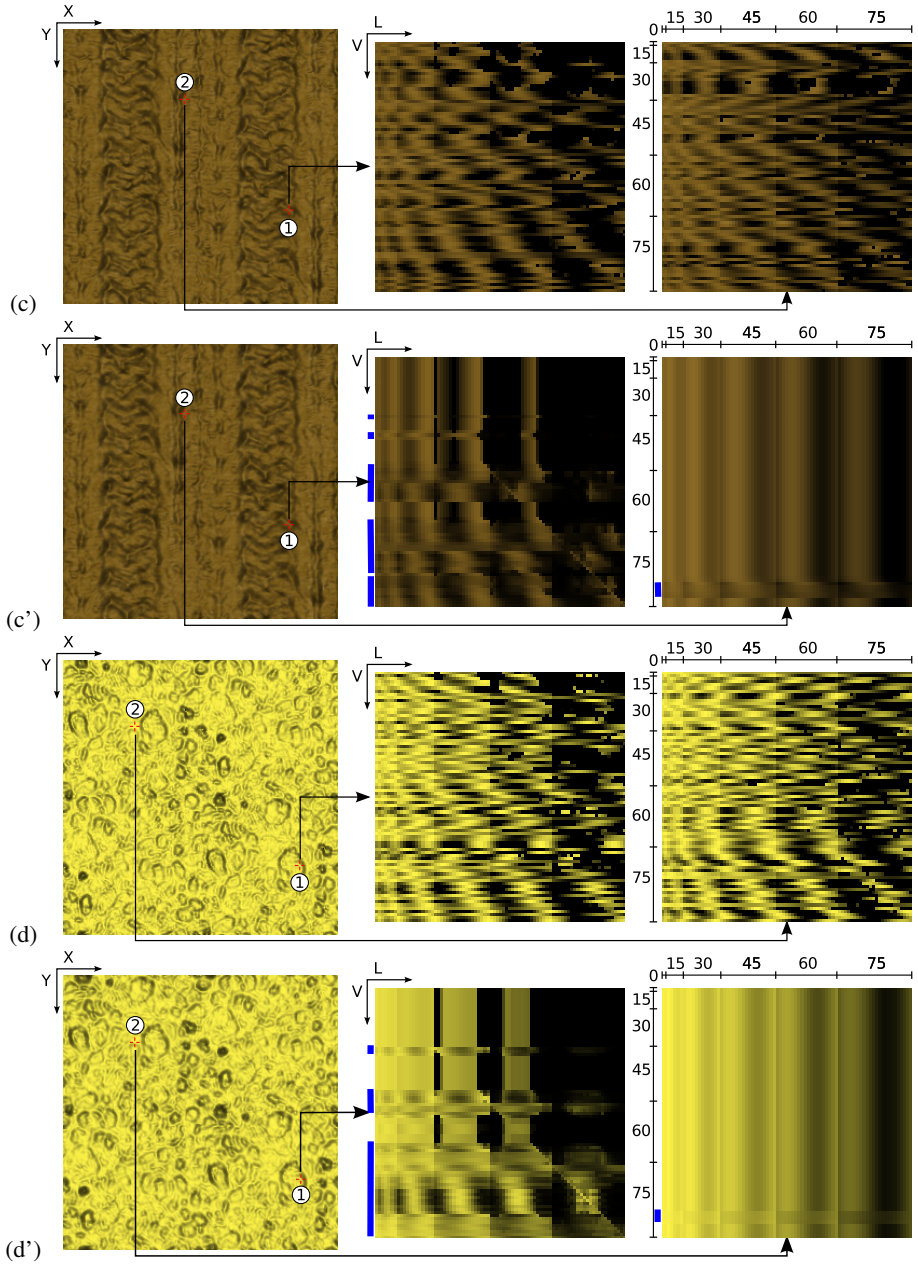


Figure 14: A visual comparison of tipical ABRDFs between a *BTF* and a *FBTF* for the *Pulli* (upper) and *Sponge* (lower) models. In (a) and (b), two ABRDFs of the normal *BTF* representation with their localisation in the spatial domain. In (a') and (b'), the corresponding ABRDFs for the *FBTF* function. For each ABRDF pair on a row, the ABRDF (1) has been picked to correspond to the same position on tne underlying meso-structure, while the ABRDF (2) corresponds exactly to the same texel in the image. The blue marks point undefined texels filled by the push-pull algorithm.

[DvNK99] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 18(1):1–34, 1999.

[FCM+08] Yannick Francken, Tom Cuypers, Tom Mertens, Jo Gielis, and Philippe Bekaert. High quality mesostructure acquisition using specularities. In *CVPR 2008: Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[FH04] Jiri Filip and Michal Haindl. Non-linear Reflectance Model for Bidirectional Texture Function Synthesis. In *Proc. International Conference on Pattern Recognition*, pages 80–83. IEEE Computer Society, 2004.

[Flo03] Michael S. Floater. Mean value coordinates. *Comput. Aided Geom. Des.*, 20(1):19–27, 2003.

[GGH02] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. *ACM Trans. Graph.*, 21(3):355–361, 2002.

[Gra03] Graphite, 2003. http://www.loria.fr/ levy/Graphite/index.html.

[HF07] Michal Haindl and Jiri Filip. Extreme compression and modeling of bidirectional texture function. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1859–1865, 2007.

[KBD07] Jan Kautz, Solomon Boulos, and Frédo Durand. Interactive editing and modeling of bidirectional texture functions. *ACM Trans. Graph.*, 26(3):53, 2007.

[KMBK03] Mellissa. L. Koudelka, Sebastian Magda, Peter N. Belhumeur, and David J. Kriegman. Acquisition, Compression, and Synthesis of Bidirectional Texture Functions. In *3rd International Workshop on Texture Analysis and Synthesis*, pages 59–64, 2003.

[LBAD+06] Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. *ACM Trans. Graph.*, 25(3):735–745, 2006.

[LHZ+04] Xinguo Liu, Yaohua Hu, Jingdan Zhang, Xin Tong, Baining Guo, and Heung-Yeung Shum. Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):278–289, 2004.

[LKG+03] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph.*, 22(2):234–257, 2003.

[MBK05] Gero Müller, Gerhard H. Bendels, and Reinhard Klein. Rapid Synchronous Acquisition of Geometry and Appearance of Cultural Heritage Artefacts. In *Proc. International Symposium on Virtual Reality, Archaeology and Cultural Heritage - VAST*, pages 13–20. EUROGRAPHICS, 2005.

[MCT+05] Wan-Chun Ma, Sung-Hsiang Chao, Yu-Ting Tseng, Yung-Yu Chuang, Chun-Fa Chang, Bing-Yu Chen, and Ming Ouhyoung. Level-of-detail representation of bidirectional texture functions for real-time rendering. In *Proc. ACM SIGGRAPH symposium on Interactive 3D graphics and games (I3D '05)*, pages 187–194, 2005.

[MGW01] Tom Malzbender, Dan Gelb, and Hans Wolters. Polynomial texture maps. In *SIGGRAPH '01*, pages 519–528. ACM, 2001.

[MK06] Sebastian Magda and David Kriegman. Reconstruction of Volumetric Surface Textures for Real-Time Rendering. In *Proc. EUROGRAPHICS Symposium on Rendering*, pages 19–29, 2006.

[MLH02] David K. McAllister, Anselmo Lastra, and Wolfgang Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. In *Proc. ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 79–88, 2002.

[MMK03] Gero Müller, Jan Meseth, and Reinhard Klein. Compression and real-time rendering of measured BTFs using local PCA. In *Proc. Vision, Modeling and Visualization*, pages 271–280, 2003.

[MMK04] J. Meseth, G. Müller, and R. Klein. Reflectance Field based real-time, high-quality Rendering of Bidirectional Texture Functions. *Computers and Graphics*, 28(1):103–112, 2004.

[MSK06] Gero Müller, Ralf Sarlette, and Reinhard Klein. Data-driven local coordinate systems for image-based rendering. *Computer Graphics Forum (Proc. EUROGRAPHICS)*, 25(3):369–378, 2006.

[MSK07] Gero Müller, Ralf Sarlette, and Reinhard Klein. Procedural Editing of Bidirectional Texture Functions. In *Proc. EUROGRAPHICS Symposium on Rendering*, 2007.

[MW08] Morgan McGuire and Kyle Whitson. Indirection mapping for quasi-conformal relief mapping. In *Proc. ACM SIGGRAPH symposium on Interactive 3D Graphics and games (I3D '08)*, 2008.

[ND06] Addy Ngan and Frédo Durand. Statistical Acquisition of Texture Appearance. In *Proc. EUROGRAPHICS Symposium on Rendering*, pages 31–40, 2006.

[NDM05] Addy Ngan, Fredo Durand, and Wojciech Matusik. Experimental Analysis of BRDF Models. In *Proc. EUROGRAPHICS Symposium on Rendering*, pages 117–226, 2005.

[PGB03] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.

[PO06] Fabio Policarpo and Manuel M. Oliveira. Relief mapping of non-height-field surface details. In *Proc. ACM SIGGRAPH symposium on Interactive 3D graphics and games (I3D '06)*, pages 55–62, 2006.

[TZL+02] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In *SIGGRAPH '02*, pages 665–672. ACM, 2002.

[VT04] M. Alex O. Vasilescu and Demetri Terzopoulos. TensorTextures: multilinear image-based rendering. *ACM Trans. Graph.*, 23(3):336–342, 2004.

[WHON97] Tien-Tsin Wong, Pheng-Ann Heng, Siu-Hang Or, and Wai-Yin Ng. Image-based Rendering with Controllable Illumination. In *Proc. EUROGRAPHICS Workshop on Rendering Techniques*, pages 13–22. Springer-Verlag, 1997.

[WWT+03] Lifeng Wang, Xi Wang, Xin Tong, Stephen Lin, Shimin Hu, Baining Guo, and Heung-Yeung Shum. View-dependent displacement mapping. *ACM Trans. Graph.*, 22(3):334–339, 2003.

Centre de recherche INRIA Bordeaux – Sud Ouest
Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex (France)

Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399